Yaoxin Li, Justin Nguyen, Vivek Rayalu

Performance Evaluation of Neural Networks on Community Detection

1 Abstract

This paper investigates the performance of three machine learning approaches for community detection on two distinct datasets: the CORA academic citation network and a Twitch user dataset. The CORA dataset presents a challenging testbed for community detection algorithms as it represents a citation network of scientific papers in computer science, while the Twitch Gamer dataset captures the mutual friend relationships between Twitch users. Our first approach uses a Multi-Layer Perceptron (MLP) that considers only node features, while the second approach uses another MLP that only considers graph data. The third approach employs a graph convolutional neural network (GCN) that combines both node features and graph data. We evaluate the performance of these models on both datasets by comparing the results to the ground-truth labels provided by the authors of the datasets. Our results demonstrate that the GCN model outperforms the two MLP models on the CORA dataset, whereas on the Twitch dataset, the MLP model that uses graph data performs better. However, further investigation is necessary to improve the performance of the GCN model that combines both graph and node features. This study emphasizes the importance of considering both node features and graph data for community detection in complex networks and underscores the potential of GCNs for this task.

2 Introduction

2.1 Definition of Community Detection

Community detection, commonly referred to as graph clustering or network clustering, is a task in network analysis which is used to identify groups or communities within a given network. These groups are usually characteristic of dense connections between nodes within the group, as compared to the connections of these nodes to the nodes in the rest of the network.

Community detection in a network is important and interesting because it can provide useful insights to the structural organization of a network that can be applied to many diverse real-world networks. Given the enormous amount of information contained in each network, the detection of communities within them would provide valuable insights and facilitate the study of the network. Moreover, the detection of communities within networks can enhance the efficiency of processing and analyzing network data. For instance, in social media, each

user represents a node, and the users' interactions with their friends create connections that form a network. Community detection algorithms can be leveraged by social media companies to identify groups of users with common friends, interests, and backgrounds, thereby improving the personalization and effectiveness of recommendation systems and advertisements. The identification of communities within a network can also provide insights into the mechanisms by which the network spreads in different contexts. Community detection has another valuable and significant application in finding missing or erroneous links within a network. By leveraging community detection algorithms, users can assign and rectify these links. Although several clustering algorithms have demonstrated good performance, they fail to incorporate additional dataset or node features. To address this limitation, we aim to investigate and compare the performance of various neural network models that leverage different features of the dataset for link prediction.

2.2 Overview of Dataset

2.2.1 CORA Dataset

The CORA dataset is a benchmark dataset used in the field of machine learning and natural language processing. It contains research papers from computer science and contains the following information: title, abstract, authors, publication venue, content, citation information. In the context of community detection, the CORA dataset can be represented as a citation network, where papers are represented as nodes and citations between papers are represented as edges. The dataset has 2708 nodes and 10,556 edges. The node features we use in our models consist of 1433 word vectors that were pre processed using natural language processing. Our models utilize node features comprising 1433 word vectors, which underwent pre-processing via natural language processing techniques. These word vectors are derived from the most frequently occurring words in all the papers and will be employed to gauge the similarities between them. This citation network can then be used to study the underlying structure of scientific communities.The papers are classified into one of seven classes which are:

1. Case Based
2. Genetic Algorithms
3. Neural Networks
4. Probabilistic Methods
5. Reinforcement Learning
6. Rule Learning
7. Theory

The CORA dataset is a widely adopted benchmark for assessing the efficacy of text classification models, graph-based machine learning algorithms, and semi-supervised learning algorithms. Given its prevalent usage in research and considerable contributions in diverse domains, we selected this dataset as the standard benchmark for evaluating the performance of neural networks on community detection.

2.2.2 Twitch Gamer Dataset

The Twitch Gamer dataset was compiled using public APIs in the spring 2018. This dataset comprises nodes representing Twitch users, while node features include attributes such as views, maturity rating, lifetime, account status, and affiliate status. The objective of our investigation was to detect communities, with the communities being defined based on the language spoken by the users. The dataset consists of 168,114 nodes and 6,797,557 edges, and is well-suited for a variety of tasks, including node regression, node classification, link prediction, and community detection.

Given the broad range of available node features, the Twitch Gamer dataset can potentially enable the development of innovative approaches for community detection that leverage diverse characteristics of the network's nodes. For instance, incorporating the lifetime feature into a neural network model may enable the identification of long-term users who are more likely to be part of stable communities, while the affiliate status feature may provide insights into the type of content that users consume and promote.

Overall, the Twitch Gamer dataset represents a valuable network to explore the effectiveness of machine learning models for analyzing social network data. By evaluating the performance of different approaches on this dataset, we can gain insights into the strengths and limitations of various machine learning algorithms, and further advance our understanding of how to leverage node features to improve the accuracy of community detection models.

2.3 Analysis of Dataset

We analyzed the degree distribution and community structure of a network, focusing on community topology. Community topology refers to the way communities are structured within a network. To determine the existence of communities, we calculated the community density, which measures the density of edges within and between communities. If a community structure exists in the graph, we expect to observe a higher density within communities than between them. The density of edges is defined by whether they connect nodes within the same community (in-community edges) or nodes in different communities (out-community edges). The in-community density is calculated based on the density of in-community edges. This approach enables us to gain insights into the community structure of a network and the patterns of connectivity between nodes. The in-community density is defined as:

$$density = \frac{number\_of\_edges\_in\_community}{number\_of\_edges\_out\_community}$$

2.3.1 Degree Distribution

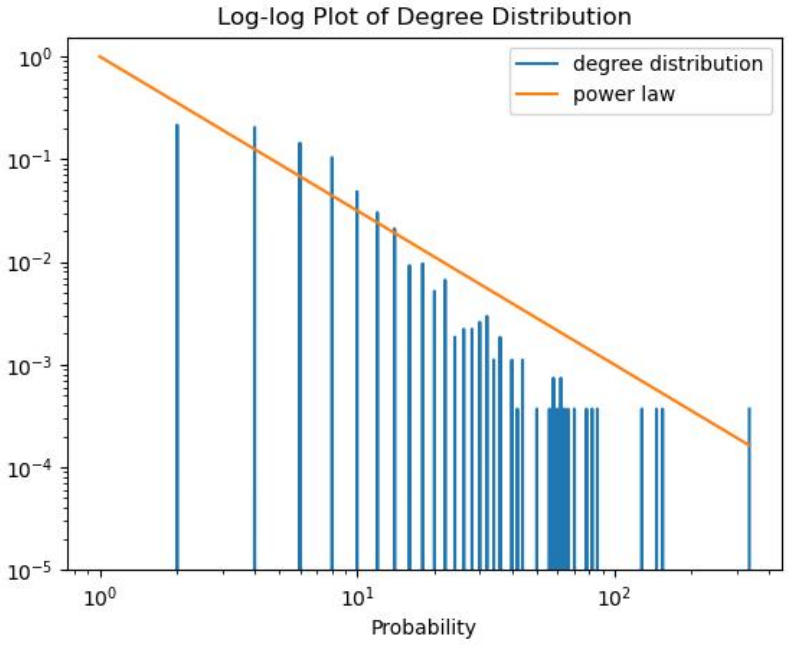

Figure 1: CORA Log Degree Distribution

As shown in Fig. 1, The degree distribution of the CORA citation network dataset exhibits characteristics of a power-law distribution, common to many real-world networks. A power-law distribution is a statistical pattern where the frequency of occurrence of events decreases

rapidly as the magnitude of the event increases. This distribution suggests the presence of hub nodes, which have significantly more connections than most nodes in the network. The majority of nodes in the dataset have a low degree, while a small number of nodes act as hubs, possessing a high degree of connections. Hub nodes are nodes with a high degree of connections. Hub nodes are critical in network analysis as they play a significant role in the network's overall connectivity and function.

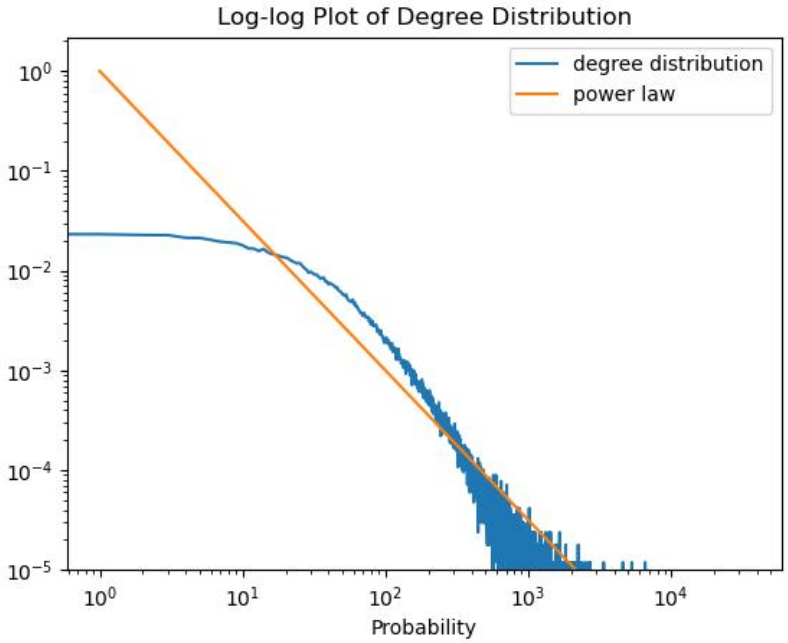

Figure 2: TWITCH Log Degree Distribution

Fig. 2 shows a plot of degree distribution for the Twitch dataset and indicates that its degree distribution follows characteristics similar to the previously mentioned CORA dataset. Specifically, the distribution of degrees in the Twitch dataset appears to follow a power-law distribution as well. As a result, the majority of nodes in the Twitch dataset have a low degree, while a small number of nodes act as hubs with a high degree of connections. T The similarity in degree distribution between the CORA and Twitch datasets suggests that both datasets have comparable network structures, where a small number of hub nodes play a significant role in the overall connectivity and function of the network.

2.3.2 Community Analysis

It is noticeable that the following terms are used for this part:

- Internal Density: Proportion of edges that exist within a community.

- Positive Density Gap: Difference between the density of edges within a community and the average density of edges outside that community. A positive density gap indicates that the community is more densely connected internally than with nodes outside the community.
- Average Density: Proportion of edges that exist within a network as a whole.
- Negative Density Gap: Difference between the density of edges within a community and the average density of edges outside that community. A negative density gap indicates that the community is less densely connected internally than with nodes outside the community.

In the CORA dataset, all communities exhibit a higher internal density, as represented by the positive density gap. The average density of these communities is relatively large, with a density gap of 0.6199. This suggests a clear community structure within the dataset in terms of community topology.

For the Twitch language communities, some communities have a negative density gap, indicating lower in-community density than out-community density. However, these communities are typically small in size. The average density of the Twitch language communities is characterized by a significant density gap of 0.799. While the English (EN) community shows a clear community structure, other communities are less significant and difficult to detect in terms of community topology. Therefore, this dataset exhibits a weak community structure in terms of community topology.

3 Methods

3.1 MLP on Node Features

The first neural network we tested on the dataset was a Multi-Layer Perceptron constructed specifically to work with purely node features. This network architecture comprised three linear layers. The first layer accepted the node feature matrix provided by the dataset, followed by Rectified Linear Unit (ReLU) activation. The second layer comprised 64 neurons as input and 32 neurons as output, which were subsequently fed into another ReLU activation function. The third and final layer consisted of 32 neurons as both input and output, and it predicted which community a node belongs to.

3.2 MLP on Graph Data

The second MLP tested on the dataset was designed to operate with a focus on graph data alone. This MLP architecture was constructed in a manner similar to the first MLP, with two linear layers and two ReLU layers of the same size. The first linear layer was followed by a ReLU activation function, while the second linear layer consisted of 64 input neurons and

32 output neurons that were subsequently fed into another ReLU layer. Finally, the third linear layer comprised 32 input neurons and generated the community prediction for a node.

3.3 GNN on Node Features + Graph Data

For this community detection algorithm, we used a Graph Convolutional Neural Network constructed to take advantage of node features and graph data. The GCN architecture comprises a Graph Convolutional Layer that accepts the node feature matrix and edge indices as input and generates 64 output channels. This layer is subsequently followed by ReLU activation, and a second Graph Convolutional Layer with 64 input channels is employed to predict the community membership of each node. The output of this layer represents the final community prediction.

4 Results

4.1 Performance Metric for the Algorithms (classification accuracy)

Each neural network was evaluated by their classification accuracy as the proportion of nodes correctly categorized to their true community divided by the total number of nodes in the data. A high classification accuracy suggests that the model performs well in predicting which community a node belongs to while a low classification accuracy implies that the model is poor at classifying which nodes belong to which community. Accuracy was chosen as the metric for its simplicity and the fact that other metrics like sensitivity and specificity work on the assumption that the prediction task is classification. Since there are many more communities than two, metrics like sensitivity and specificity are not viable for our analysis.

4.1.1 Model Performance on CORA dataset

The models' performance on the CORA dataset is displayed in Table 1. below:

Model	Accuracy
Multi-Layer Perceptron on Node Features	77.12%
Multi-Layer Perceptron on Graph Data	72.69%
Graph Convolutional Neural Network	90.04%

Table 1: Results of CORA dataset

The results of the models on the CORA dataset demonstrate that the GCN incorporating both node features and graph topology resulted in the highest classification accuracy when compared to the other two models evaluated. The other two models yielded sub-par accuracies of around 75%. These results were expected since the CORA dataset has relevant node

features and a well-defined community structure which makes it well suited for community detection. More specifically, since the relevant node features can be utilized independently for community detection, we anticipated that the MLP model employing node features would yield somewhat satisfactory classification accuracy. We also expected that the MLP model using graph structure would provide similar performance. Hence, since both node features and graph topology contribute to predictive accuracy, the GCN model is naturally capable of delivering superior performance relative to the other two models.

4.1.2 Model Performance on Twitch dataset

The models' performances on the Twitch dataset is shown in the table below:

Model	Accuracy
Multi-Layer Perceptron on Node Features	74.16%
Multi-Layer Perceptron on Graph Data	89.10%
Graph Convolutional Neural Network	74.08%

Table 2: Results of Twitch dataset

Unlike what we observed for the CORA dataset, the results from the Twitch Gamer dataset are counter intuitive. The model with the best performance is the model of MLP with graph structure. Despite the GCN using node features and graph topology, it does not lead to a better classification accuracy. Although we did implement adaptive learning rate to avoid over-fitting the models, it is still possible to over-fit. The findings from the analysis of the Twitch Gamer dataset provide valuable insights. The data revealed that the EN (English) language community constitutes approximately 74% of the dataset which makes sense since most twitch streamers stream in English. The MLP model which uses node features and the GCN model had similar accuracies of around 74%. It is possible that these two models were unable to fit the data and only generate random predictions. Upon analyzing the dataset, we identified that the node features including views, maturity, lifetime, account status, and affiliate may not be closely related to the community, which could explain the inability of the models to fit the data. The MLP model with graph structure demonstrated the best performance, achieving 89.10% accuracy, which shows that using solely graph structure might be more suitable depending on the type of network. Therefore, it is likely that utilizing node features negatively impacted the performance of the GCN model, despite its incorporation of graph structure information.

5 Discussion

5.1 Strengths and Weaknesses of each Algorithm

5.1.1 MLP on Node Features

The Multi-Layer Perceptron working solely based on node features makes it a simple algorithm to utilize for community detection. The model does not need a set of nodes and edges to deploy, so it is able to perform in the absence of graph data. However, its simple dependence on the node features means if the node features are not predictive of which community a node belongs to, the model is bound to perform poorly. Also, the model can only perform so well in the absence of further information like the nodes and edges of the graph data.

5.1.2 MLP on Graph Data

The strengths of using a MLP for community detection using only graph data are its simplicity and flexibility. MLPs are straightforward models that can easily handle simple linear relationships between input and output data, making them well-suited for processing graph data. They are also flexible and can be easily modified to accommodate different types of graph data. Additionally, MLPs can generalize well to unseen data, making them well-suited for community detection tasks where predictions need to be made on new, previously unseen nodes. The weaknesses of using an MLP for community detection include limited model capacity, the potential for overfitting, and insensitivity to graph structure. MLPs are limited in their ability to capture complex non-linear relationships between input and output data, which can result in suboptimal performance on large and complex graphs.

5.1.3 GNN on Node Features + Graph Data

The graph convolutional neural network makes use of both the node features and set of nodes and edges given in the data to predict communities. The model depends on the existence and quality of both the node features and graph data, so having both is a prerequisite to implement this neural network. To its advantage, it has more potential to perform well precisely because it can use more data to make predictions. A disadvantage of the GCN is that it can result in flawed predictions when the node features are not closely related to the underlying communities they belong to. Another disadvantage can occur when the community structure of a network is indistinct, indicated by a low or negative community density difference. In such cases, the graph structure does not facilitate community detection for traditional clustering algorithms or the GCN. However, GCNs can leverage node features to optimize the model, which gives them an edge over traditional algorithms.

5.2 Comparison of Results

5.2.1 CORA dataset

A comparison of the results of three models for community detection on CORA dataset shows that the GCN performed the best, achieving an accuracy of 90.04%. The MLP which used node features had the second-best performance, with an accuracy of 77.12%. However, the MLP which used graph data showed a lower accuracy of 72.69%. These results indicate that incorporating node feature information into the model can lead to improved performance in community detection tasks. The GCN, which was specifically designed to process node features and graph data, demonstrated the best results, while the MLP on Node Features showed a significant drop in accuracy when only using node features. These results highlight the importance of considering both node features and graph structure information in community detection tasks.

5.3 Twitch dataset

The analysis of the Twitch Gamer dataset revealed that node features can sometimes have a detrimental effect on the predictive capacity of the models. While the results from the CORA dataset suggest that incorporating both graph structure and node features can improve performance, the findings from the Twitch dataset indicate that irrelevant node features can reduce the predictability of the models. As previously mentioned, the node features in the Twitch Gamers dataset are probably not directly related to language communities, and including these extraneous features in the models substantially diminishes their predictive power.

6 Conclusion

6.1 Summary of Findings

The GCN model which used graph data and node features worked best at detecting communities on the CORA dataset, and, thus, we can get following conclusions:

- Using both components of the data led to a substantial improvement in accuracy, when both graph structure and node features are predictive.

The MLP on solely the graph structure performed the best on the Twitch Gamers dataset, and, hence, we can say that:

- Using node features could have brought the accuracy down, and since the MLP on node features and GCN incorporated it, they had lower accuracy. Therefore, the node features are not as predictive as the graph data on the twitch dataset, and node features mislead the model.

We can conclude that incorporating both graph structure and node features in a model, when both are predictive, would result in improved performance. This is because the model can leverage more predictive information, which would naturally improve its performance. However, if the information provided to the model is unhelpful or misleading, it would not contribute to its performance. All in all, the best performing model depends on the quality of the data, so it is important to be flexible and use whichever is best for the situation.

6.2 Recommendations and Implications for Further Research

So far, the models included in this analysis were assumed to work on graph datasets where the edges were undirected, unweighted, and had no features. Assuming and going off of the result that the neural network that employed more data performed better, it may be interesting to further look into either creating new models or complicating our graph convolutional neural network that can account for datasets with directed and undirected edges, edge weights, and edge features to better improve predictions. A possible place to start could be inputting edge weights into the graph convolutional layers included in our graph convolutional neural network since edge weights are an optional input to feed into the layer, suggesting it should be possible to work with edge weights. Then from there, dive deeper into bringing in more data like edge directions and features.

7 Reference

Hagberg, A. A., Schult, D. A., Swart P. J. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX in Proceedings of the 7th Python in Science conference (SciPy 2008): 11-15. https://conference.scipy.org/proceedings/SciPy2008/paper$_2$/$full_text.pdf$

Kipf, T. N., Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks. arXiv. https://doi.org/10.48550/arxiv.1609.02907

Paszke, A. et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems 32: 8024-8035. Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Yang, Z., Cohen W. W., Salakhutdinov R. (2016). Revisiting Semi-Supervised Learning with Graph Embeddings. Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: WCP volume 48. arXiv. https://doi.org/10.48550/arXiv.1603.08861